5ZC-02

複数のハードウェアでの共通操作に着目した教育用並列プログラミング言語の提案

田中　寛章‡　　藤井　健太†　　礒淵　郁也†　　水谷　泰治†

大阪工業大学　情報科学研究科‡、　大阪工業大学　情報科学部†

1. はじめに

近年，マルチコア CPU や GPU といった並列計算可能なハードウェアが普及したことで，並列計算はますます注目されている．一方で，並列プログラムを記述する場合，ハードウェアにより別々の並列プログラミング言語やライブラリを用いる必要がある．これらはそれぞれ仕様や記述法が独特であり学習者にとって難しい．

本稿では，並列プログラミング言語やライブラリの共通操作と記法を抽出した教育用並列プログラミング言語を提案する．教育用並列プログラミング言語で並列プログラムを記述し，実行することで，既存の並列プログラミング言語やライブラリを学ぶ際の予備知識となる．

2. 並列プログラミングについて

並列計算を行えるハードウェアには PC クラスタ，マルチコア CPU，GPU 等がある．これらに対応した並列プログラミング言語やライブラリにはそれぞれ MPI[1]，Pthread，CUDA[2]がある．図 1，図 2 にそれぞれ配列 A，B の同じ添字の値の和を配列 C に格納するプログラムを示す．図中のコメントの通り，環境によって書き方が異なり，この点が並列プログラミングを学習する際の障壁となっている．

3. 教育向け並列プログラミング言語の提案

学習者が並列プログラミングを行うだけなら，OpenMP や CilkPlus のようなコーディングを単純化するツールがある．しかし，ハードウェアの性能を引き出すには，ハードウェアに合わせたコーディングを行うのが有利である．そのため，学習者は将来的にハードウェアに合わせた並列プログラムをコーディングできるようになることが望ましい．

```
#include <mpi.h>
#define N 4
MPI_Status s;
int A[N],B[N],C[N],tag =100;
int main (int argc,char *argv[]){
 int id, c, i;
 MPI_Init(&argc,&argv);
 //プロセス ID と合計数を取得
 MPI_Comm_rank(MPI_COMM_WORLD,&id);
 MPI_Comm_size(MPI_COMM_WORLD,&c);
 //プロセス毎に違う領域を処理させる
 for(i=(id*N)/c;i<((id+1)*N)/c;i++)
  C[i]=A[i]+B[i];
 if(id==0)
  for(i=1; i<c; i++)
   MPI_Recv(C+(N*i),N,MPI_INT,i,…);
  else
   MPI_Send(C,N,MPI_INT, 0, tag, …);
 MPI_Finalize();
}
```

図 1　MPI のプログラム

```
#include <pthread.h>
#define N 4
#define t 4
int A[N],B[N],C[N];
//並列処理させる動作を関数で宣言
void *f( void *ptr){
 int i,id = (*(int*)(ptr));
 for(i=id*N/t; i<((id+1)*N)/t;i++)
  C[i]=A[i]+B[i];
 return NULL;
}
int main(int argc ,char *argv[]){
 pthread_t th[t];
 int n[t], i;
 for(i=0;i<id;i++){
  n[i] = i;
  int *s = &n[i];
  //関数 f を別スレッドで実行する
  pthread_create(&th[i],NULL,f,s);
 }
 for(i=0;i<t;i++)
  pthread_join(th[i],NULL);
 return 0;
}
```

```
#define N 4
#define t 4
//並列処理させる動作を関数で宣言
__global__voidf(int*A,int*B,int*C){
 int id = blockIdx.x * blockDim.x +
threadIdx.x;
 for(i=(id*N)/t;i<(id+1)*N/t;i++)
  C[id] = A[id] + B[id];
}
int main(void){
 int A[N],B[N],C[N],*ad,*bd,*cd;
 //GPU 上のメモリ領域を確保する
 cudaMalloc(&ad, sizeof(int)*N);...
 cudaMemcpy(ad,a,sizeof(int)*N,
  cudaMemcpyHostToDevice);...
 //関数 f を並列実行する
 f<<<2, 2>>>(ad, bd, cd);
 cudaDeviceSynchronize();
 cudaMemcpy(C,cd,sizeof(int)*N,
  cudaMemcpyDeviceToHost);
 //GPU 上のメモリを開放する
 cudaFree(ad)...;
 return 0;
}
```

図 2　Pthread(左)，CUDA(右)のプログラム

そこで，本研究では，MPI，Pthread，CUDA の独特な部分を排除し，これらの共通した操作を抽出し，その共通操作のみで並列プログラムを記述・実行できる教育用並列プログラミング言語(以下 T 言語)を提案する．T 言語を用いて並列プログラミングを学習する事で，どの環境で並列プログラミングを学習する際でも予備知識となり，学習負荷を低減できると考えられる．

さらに分散・共有メモリ型における共通操作についても抽出する．メモリ型によって操作は異なるが同じメモリ型の場合は共通している操作が存在する為，それぞれ仕様に加える．

また，並列プログラミングの学習が難しい理由の 1 つに，実行状況が把握しづらいことが挙げられる．並列プログラムの動作への理解を促進するため，T 言語で記述したプログラムを実行し，結果を確認するためのエミュレーションと，動作の分かりづらい処理の可視化を行う．

4. 教育向け並列プログラミング言語の詳細

T 言語で着目した共通操作について述べる．

4.1 抽出した共通操作

図 3 に抽出した操作を関数の形で表す．

並列実行を行う T_Run 関数を定義した．並列実行は Pthread では pthread_create，CUDA では

An Educational Parallel Programming Language based on Common Operation for Multiple Hardware.

†Hiroaki Tanaka, Kenta Fujii, Fumiya Isobuti, Yasuharu Mizutani

‡Graduate School and Faculty of Information Science and Technology

†Faculty of Information Science and Tchnology, Osaka Institute of Technology

void T_Run(int num , void (*f)(void));
関数*f を並列実行数 num で並列実行する.
int T_GetMyNum(void);
呼び出したプロセス/スレッドの通し番号を取得する.
int T_GetTotalNum(void);
現在の並列実行数を取得する.
void T_Barrier(void);
並列処理中の各処理を停止し，実行中の他の処理が終了するまで待つ.
void T_Send(int sendNum, void *value, int size);
アドレス value の内容を size 分だけ通し番号 sendNum に送信する.
void T_Recv(int RecvNum, void *value, int size);
T_Send 命令で送信される size 分データをアドレス value に格納する.
void T_Lock(int lockNum);
void T_UnLock(int unLockNum);
同じ lockNum で指定された T_Lock 以降の処理を，T_UnLock が実行されるまで他の処理は実行せず待機させる.

図 3　抽出した共通操作

特殊記号を用いて関数を並列呼び出しする．MPI では mpiexec でプログラムを並列実行するが，これは main 関数を並列実行していると解釈できる．並列実行数はどの環境でも並列実行時に指定している.

並列実行中で違う処理を割り当てる場合，通し番号を用いる．MPI では MPI_Comm_rank 関数，Pthread ではスレッド毎に違う値を引数に与える．CUDA では組み込み変数を用いて通し番号を取得する．同様に，並列実行数も別々の処理を割り当てるのに必要な情報である．これらの情報を取得する関数を T_GetMyNum 関数，T_GetTotalNum 関数とした.

プログラムにより実行の順序を制御する必要がある．T_Barrier 関数を実行することで実行中の処理の足並みを揃えることができる．MPI，Pthread，CUDA でも同等の関数が用いられている.

分散メモリモードの場合，各処理が計算したデータを 1 ヵ所に纏める必要があり，MPI では通信を行うプロセスの番号を指定して送受信する．T 言語でも同様に T_Send 関数と T_Recv 関数を用いて通信を行う.

排他制御は T_Lock 関数，T_UnLock 関数で行う．共有メモリモードにおいて，同じメモリ番地に同時にアクセスすると処理結果が正しくない場合があるため，排他制御を行い一方がアクセス中の場合は他の処理は待機させる必要がある.

4.2 エミュレート

プログラムを既存の並列プログラミング言語やライブラリを用いたプログラムに変換し，実行する．分散メモリモードの場合は MPI，共有メモリモードの場合は Pthread に変換した．変換方法を図 4 に示す．共有メモリモードの場合は共有メモリモード用ランタイムファイルと共にコンパイルすることで T 言語関数は Pthread 関数のラッパーとして機能する．分散メモリモードも同様に分散メモリモード用ランタイムファイルとコンパイルするが，ソースファイル内の T_Run 関数呼出しを抽出し，並列実行数を mpiexec で並列実行するプロセス数とし，T_Run 関数は並列実行関数の呼出しに置き換える．さらに，main 関数を T_main 関数とし予めランタイム内で定義されている，MPI の初期化処理等が記述された main 関数内で呼び出す.

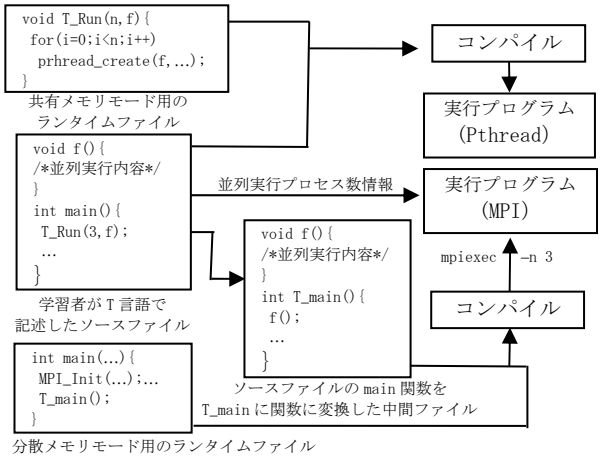

図 4 T 言語から Pthread,MPI への変換

4.3 可視化

並列実行中の処理を可視化する．本稿では，学習者にとって動作のつかみにくい分散メモリ型におけるデータ通信処理を可視化した．図 5 は，3 つの関数を T_Run 関数で並列実行し，それらの送信と受信の処理を図示したものである．横軸は時間の経過を表しており，プロセス間にひかれた線は成立した通信処理を表している.

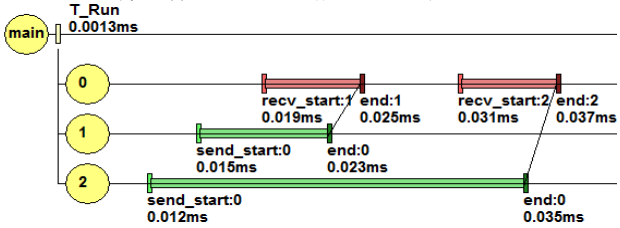

図 5　分散メモリ型における通信処理の可視化

5. 動作確認

T 言語で分散メモリ型ガウスの消去法，共有メモリ型ヒストグラムのプログラムを記述し，変換を行った．変換後実行し，正しい計算が行われている事を確認した.

6. まとめと今後の課題

本稿では，教育用並列プログラミング言語を提案した．そして，その言語で記述したプログラムのエミュレートと可視化機能を実装した．今後の課題として，並列プログラミング学習者への評価実験が挙げられる.

謝辞

本研究は科学研究費補助金若手研究(B)(15K21511)の補助による.

参考文献

[1] http://www.mpi-forum.org/
[2] https://developer.nvidia.com/cuda-zone/